RSAC 2025 Conference

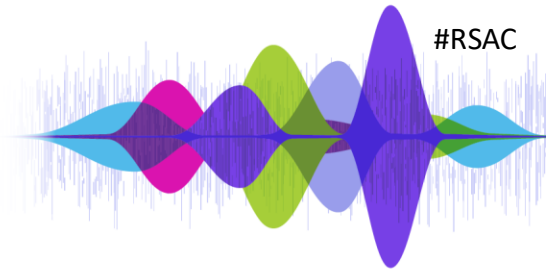
Many Voices.
One Community.

SESSION ID: CLS-M01

Securing Cloud Access with Kubernetes Workload Identity

Eric Johnson

Principal Security Engineer, Puma Security
Senior Instructor, SANS Institute
linkedin.com/in/eric-m-johnson/



Disclaimer

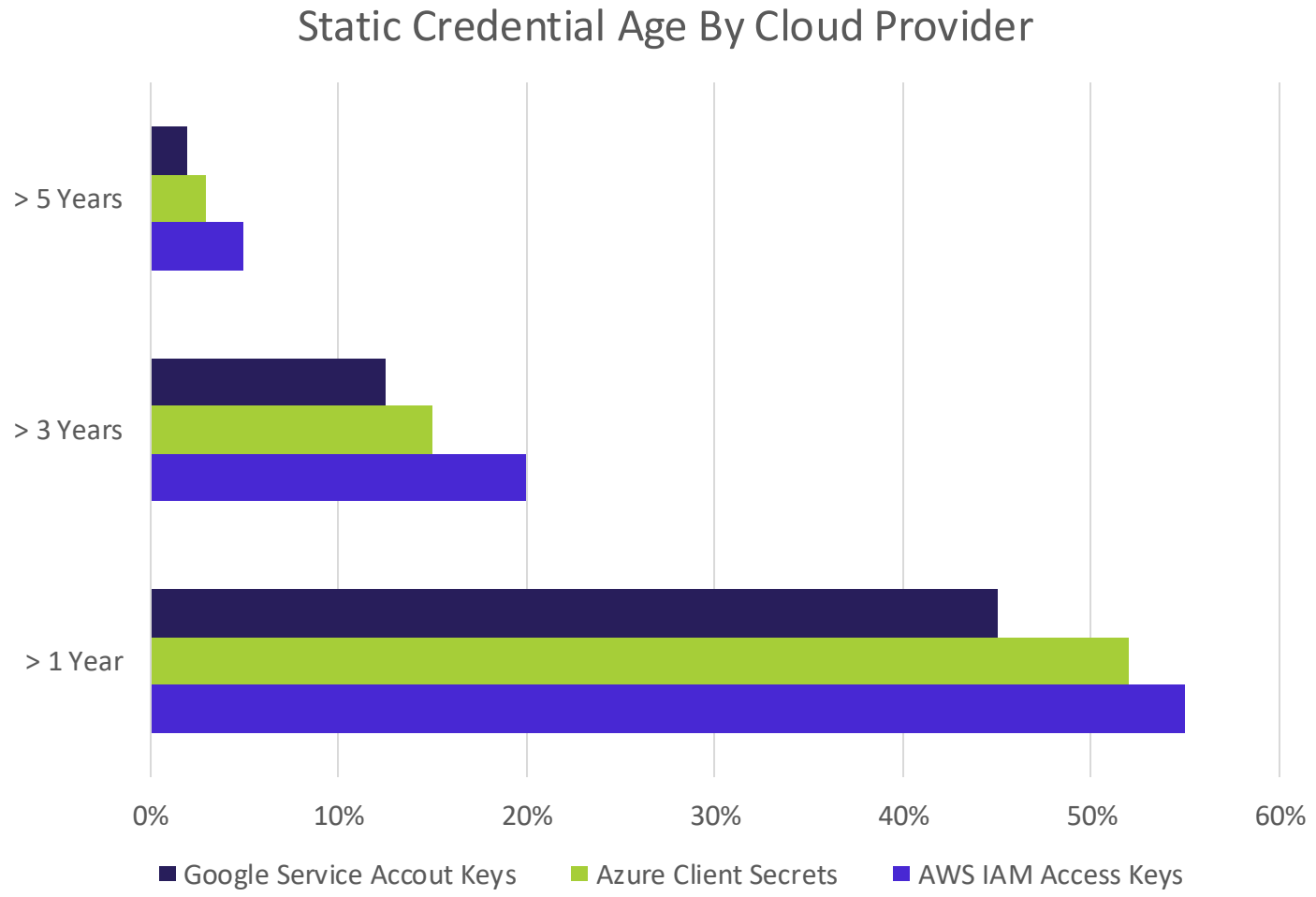
Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

Cloud Attack Techniques

- Static credentials primarily support programmatic scripts and applications
- Rotating static credentials is challenging, so we avoid it
- Lost or stolen static credentials are the initial access method in ~ 66% of cloud breaches
- Sources:
  - AWS re:Inforce TDR432: New tactics and techniques for proactive threat detection
  - *2024 Datadog State of Cloud Security*



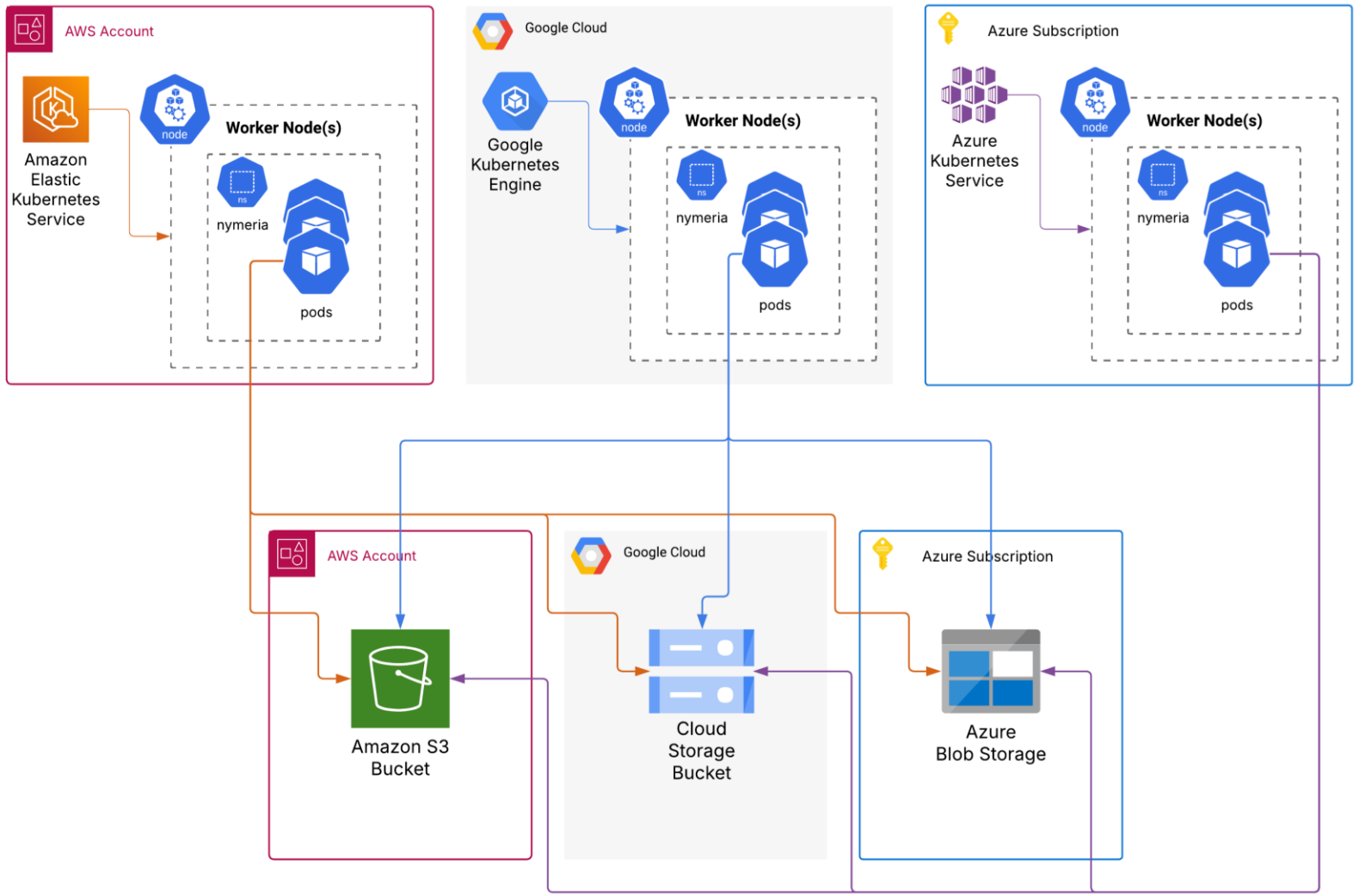
Workload Identity GitHub Repository

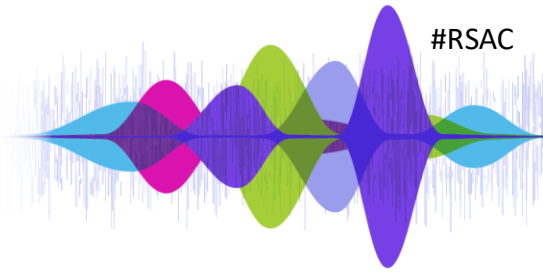
Nymeria is an open-source repository with many cross-cloud static credential and workload identity scenarios:

- Kubernetes
  - AWS EKS, Azure AKS, GKE
- Serverless functions
  - AWS Lambda, Azure & Google Cloud Functions
- GitHub Actions / Azure virtual machines

https://github.com/pumasecurity/nymeria

The Problem: Kubernetes Cross Cloud Authentication





Session Goals

- Review Kubernetes pod credential anti-patterns
- Learn how to enable Kubernetes workload identity
- Configure pod service account identity tokens for intra-cloud access to resources
- Extend pod service account identity tokens to meet external cloud provider audience and expiration requirements
- Programmatically authenticate from a pod to AWS, Azure, and Google Cloud APIs

RSAC 2025 Conference

Kubernetes Pod Credential Anti-Patterns

Securing Cloud Access with Kubernetes Workload Identity

Many Voices.
One Community.

Cloud Provider Static Credential Types

Each cloud provider's Identity & Access Management (IAM) service provides an option for creating static, long-lived credentials:

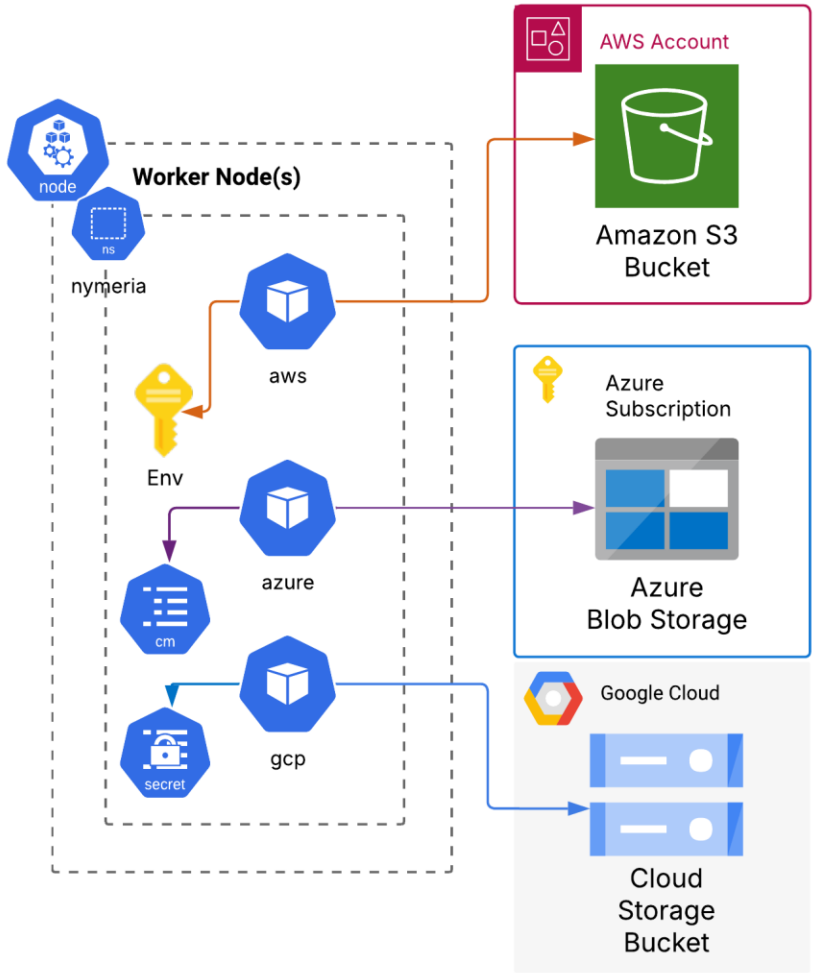
IAM User Access Keys

Service Principal Client Secrets

Google Service Account Keys

Pod Credential Anti-Patterns

- Generating static, long lived credentials for accessing cloud provider APIs
- Storing static cloud credentials in Kubernetes objects:
  - ConfigMap
  - Secrets
  - Annotations
- Provisioning credentials to pods through environment variables or file system volume mounts



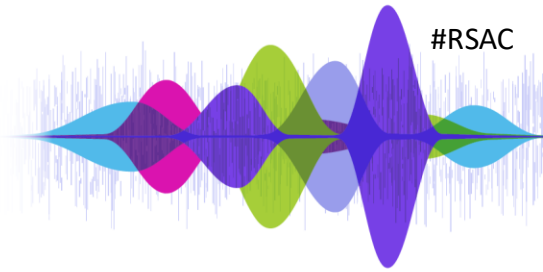
Kubernetes Pod ConfigMap Mount

Discovering static credentials provisioned using a *ConfigMap*:

```
$ kubectl describe pods -n static-credential -l cloud=gcp

Name:             nymeria-gcloud-8478f4cb8f-bs8kr
Namespace:        static-credential
…
Containers:
  gcloud:
    …
    Image: gcr.io/google.com/cloudsdktool/google-cloud-cli:latest
    Mounts:
      /mnt/gcp/sa from gcp-sa-key (rw)
…
Volumes:
  gcp-sa-key:
    Type:       ConfigMap (a volume populated by a ConfigMap)
    Name:       gcp-sa-key
```

Kubernetes ConfigMap Exfiltration



Exfiltrating *ConfigMap* data containing static credentials:

```
$ kubectl describe configmaps -n static-credential gcp-sa-key

Name:         gcp-sa-key
Namespace:    static-credential

Data
====
credentials.json:
----
{
      "type": "service_account",
      "project_id": "your-project",
      "private_key": "-----BEGIN PRIVATE KEY…"
      "client_email": "sa@your-project"
      …
}
```

Kubernetes Pod Environment Variables

Discovering static credentials provisioned in a pod's environment variables:

```
$ kubectl describe pods -l cloud=aws

Name:              nymeria-aws-58fcfc8969-t7qwr
Namespace:         static-credential
Containers:
  awscli:
    Container ID:  containerd://…
    Image:          amazon/aws-cli:latest
    …
    Environment:
      AWS_ACCESS_KEY_ID: <set to the key
      'aws_secret_access_key_id' in secret 'aws-iam-user'>
      AWS_SECRET_ACCESS_KEY: <set to the key
      'aws_secret_access_key' in secret 'aws-iam-user'>
```

Kubernetes Secrets Exfiltration

Exfiltrating secret values containing static credentials:

```
$ kubectl get secrets -n static-credential -o json aws-iam-user

{
  "apiVersion": "v1",
  "data": {
    "aws_secret_access_key":
      "SElacHZmSllCU3M3cWNyZ3ZZYURUSnJMcnE0S2llMGQ5aDFYcG5MTg==",
    "aws_secret_access_key_id": "QUtJQVNaWTJaU1U2WVVFWE5HSTY="
  },
  "kind": "Secret",
  …
  },
  "type": "Opaque"
}
```

Kubernetes Credentials Mitigations

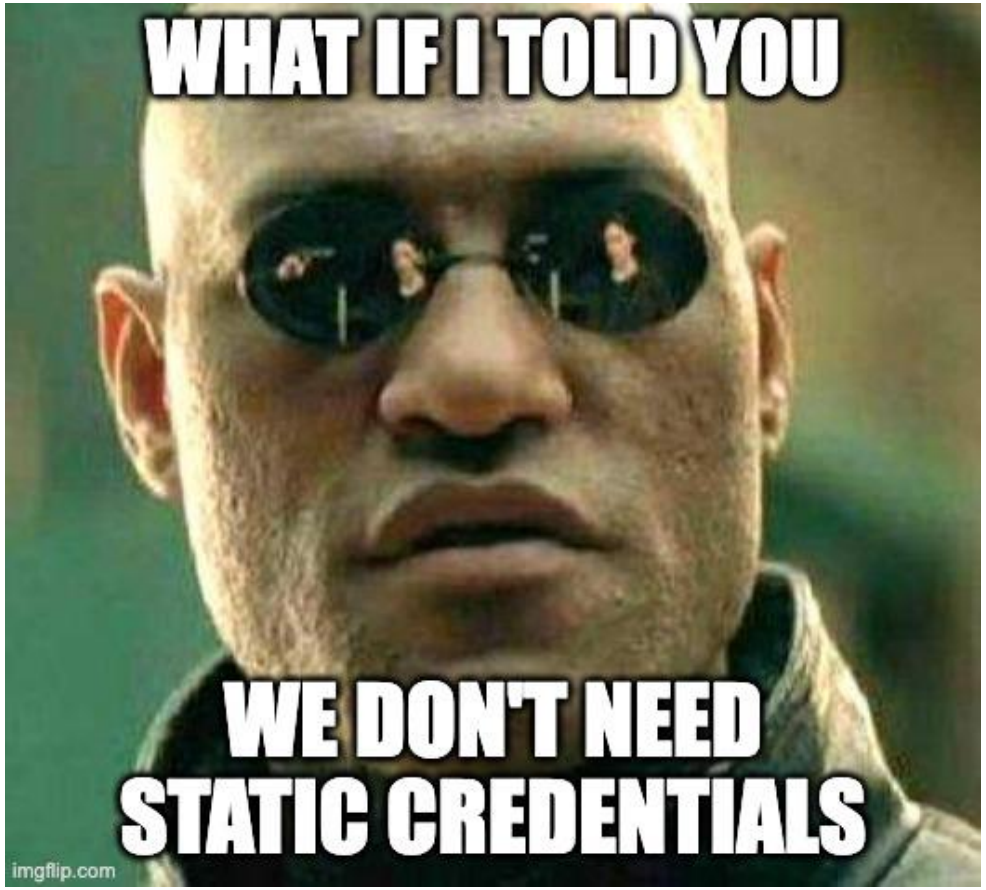
Common mitigations….

- Version control secrets scanning

- Kubernetes secrets management integrations

- Compromised credential detection

But….



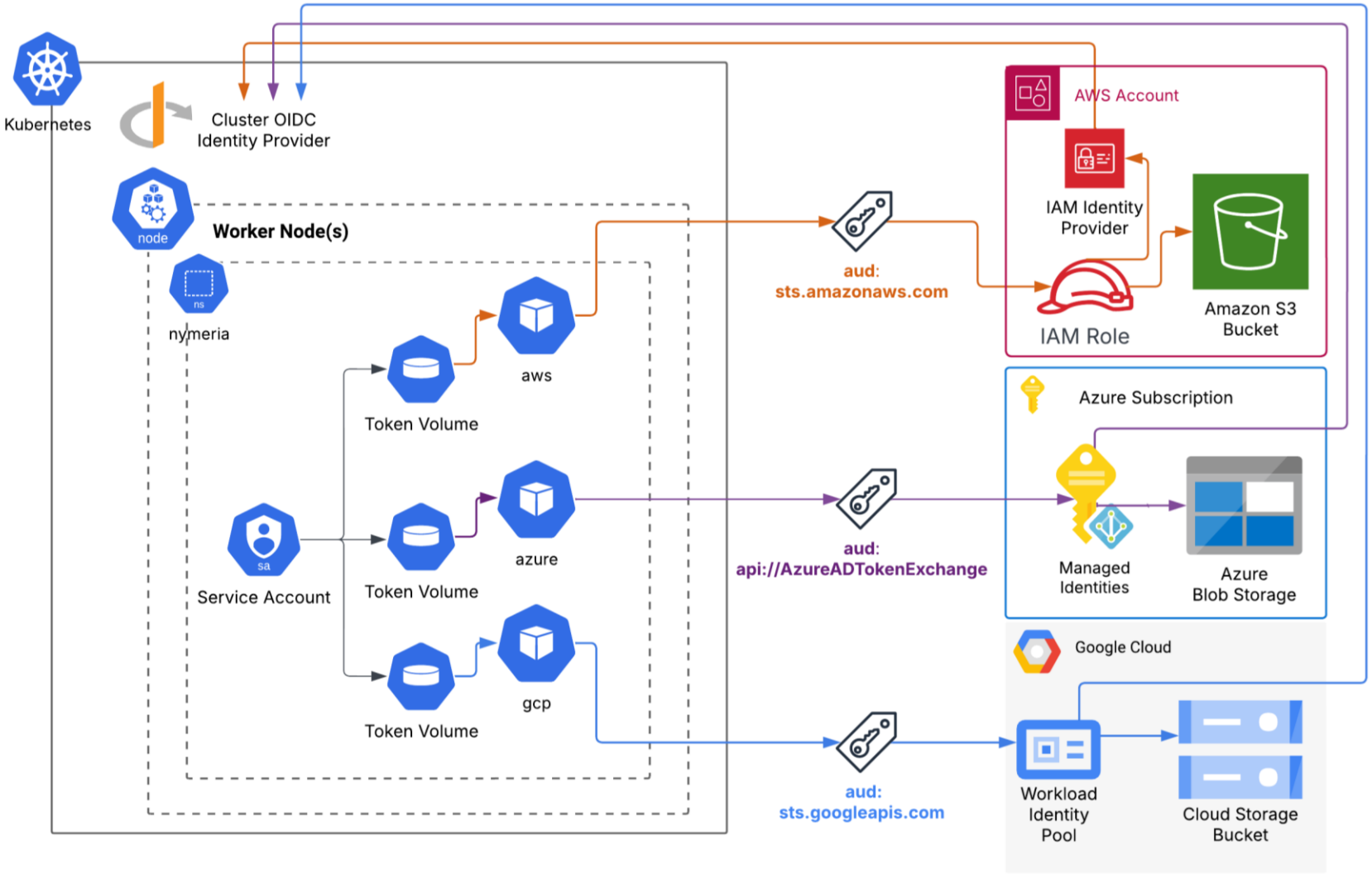
RSAC 2025 Conference

Kubernetes Workload Identity Federation

Securing Cloud Access with Kubernetes Workload Identity

Many Voices.
One Community.

Kubernetes Workload Identity Federation



Cloud Kubernetes Service Capabilities

Cloud managed Kubernetes service add-ons enable workload identity for pods accessing the cloud provider APIs:

AWS Elastic Kubernetes Service (EKS)	Create an IAM Roles for Service Accounts (IRSA) OIDC identity provider or install the EKS Pod Identity add-on
Google Kubernetes Engine (GKE)	Enable workload identity and create a namespace for Kubernetes service accounts (automatically enabled for auto pilot)
Azure Kubernetes Service (AKS)	Enable the workload identity and OIDC issuer add-on options for the cluster

Kubernetes Workload Identity Configuration

Kubernetes workload identity allows pods to securely access cloud provider APIs without managing static credentials:

1. Create a Kubernetes service account
2. Set required service account annotations (varies by cloud provider)
3. Assign the service account to the pod
4. Observe the service account identity tokens are automatically fed into the pod and rotated by the cluster

GKE / Google Cloud Workload Identity

- Kubernetes *ServiceAccount* resources provide a unique identity for workloads running inside a namespace
- Deployments (or pods) using the *serviceAccountName* will have a signed OIDC identity token projected into a volume mount
- GKE specific service account annotations are ***not*** required

```
1  ---
2  apiVersion: v1
3  kind: ServiceAccount
4  metadata:
5    name: "nymeria"
6    namespace: workload-identity
7  ---
8  apiVersion: apps/v1
9  kind: Deployment
10 …
11 spec:
12   …
13   template:
14     …
15     spec:
16       serviceAccountName: nymeria
17       containers:
18         - name: gcloud
```

GKE Workload Identity Tokens

Service account identity tokens are automatically mounted into a volume for the pod to use for intra-cluster and Google cloud API access:

```
$ kubectl exec -n workload-identity -it $(kubectl get pod -n workload-
identity -l cloud=gcp -o json | jq -r '.items[0].metadata.name') --
cat /var/run/secrets/kubernetes.io/serviceaccount/token
```

Decoding the service account identity token reveals the GKE cluster's default issuer and audience, as well as the node, pod, and service account:

```
{
  "aud": [ "https://container.googleapis.com/v1/projects/my-project
           /locations/y-region/clusters/nymeria" ],
  "iss": "https://container.googleapis.com/v1/projects/my-project
          /locations/my-region/clusters/nymeria",
  "sub": "system:serviceaccount:workload-identity:nymeria"
  "kubernetes.io": {…} …}
```

GKE Workload Identity Policy Binding

GKE service accounts can be directly added to project / resource level IAM permission bindings:

- GKE automatically creates a workload identity pool in the project
- Example: granting a GKE service account principal the storage object viewer role on a bucket:

```
$ gcloud storage buckets get-iam-policy gs://nymeria-123456

bindings:
  role: roles/storage.objectViewer
  - members:
    - principal://iam.googleapis.com/projects/########/locations/
      global/workloadIdentityPools/my-project.svc.id.goog/subject
      /ns/workload-identity/sa/nymeria
```

AWS EKS IRSA / AKS Workload Identity

EKS *ServiceAccount* resources use annotations to configure workload identity:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "nymeria"
  namespace: workload-identity
  annotations:
    eks.amazonaws.com/role-arn:
      arn:aws:iam::012345678901:
      role/nymeria
    eks.amazonaws.com/token-
    expiration: "3600"
```

AKS *ServiceAccount* resources use annotations to configure workload identity:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "nymeria"
  namespace: workload-identity
  annotations:
    azure.workload.identity/
    client-id: your-client-id
    azure.workload.identity/
    tenant-id: your-tenant-id
    azure.workload.identity/
    service-account-token-
    expiration: "3600"
```

AWS EKS IRSA Identity Tokens

Service account identity tokens are automatically mounted into a volume for the pod to use for AWS API authentication:

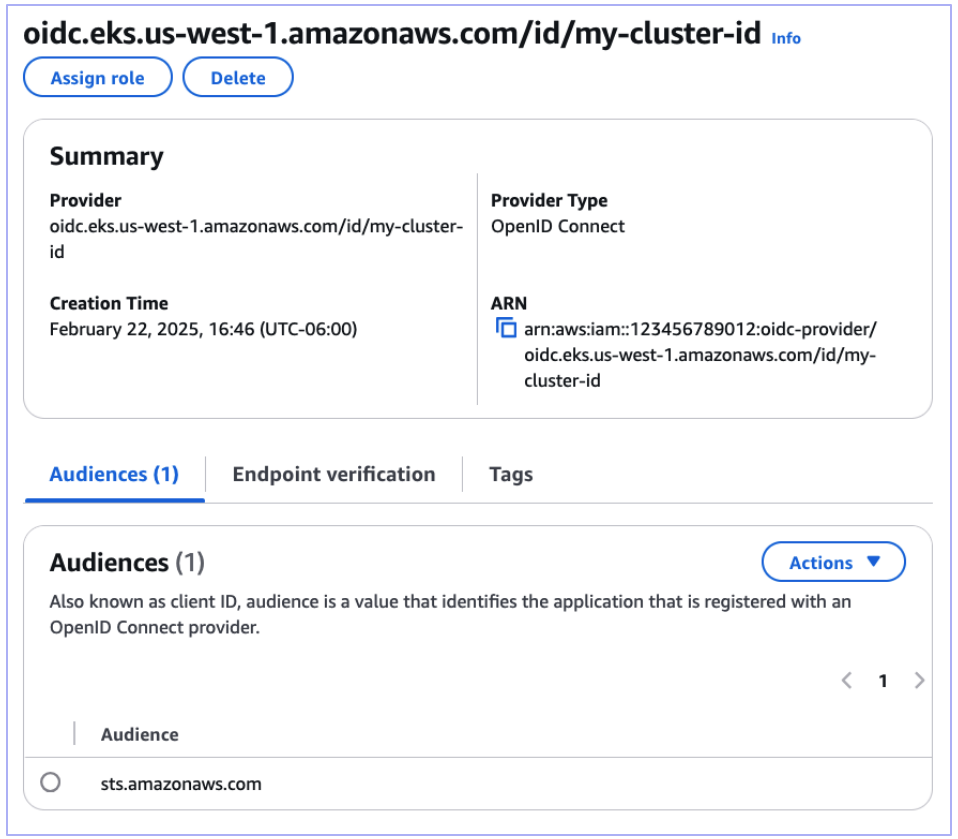
```
$ kubectl exec -n workload-identity -it $(kubectl get pod -n workload-
identity -l cloud=aws -o json | jq -r '.items[0].metadata.name') --
cat /var/run/secrets/eks.amazonaws.com/serviceaccount/token
```

Decoding the service account identity token reveals the EKS cluster's default issuer and audience, as well as the node, pod, and service account:

```
{
  "aud": [ "sts.amazonaws.com" ],
  "iss": "https://oidc.eks.us-west-1.amazonaws.com/id/my-cluster-id",
  "sub": "system:serviceaccount:workload-identity:nymeria"
  "kubernetes.io": {…}
  …
}
```

AWS EKS IRSA Role Assumption

1. AWS Identity Provider resource trusting identity tokens issued from the EKS cluster OIDC identity provider:



2. AWS IAM Role trust policy granting the EKS pod's service account permissions to assume the role:

```
{ "Effect": "Allow",
  "Principal": {
    "Federated":
      "arn:aws:iam::123456789012:
       oidc-provider/oidc.eks.us-west-1
       .amazonaws.com/id/my-cluster-id"
  },
  "Action":
    "sts:AssumeRoleWithWebIdentity",
  "Condition": {
    "StringEquals": {
      "oidc.eks.us-west-1.amazonaws.com
      /id/my-cluster-id:sub":
      "system:serviceaccount:workload-
      identity:nymeria"
} }
```

Azure Kubernetes Workload Identity Tokens

Service account identity tokens are automatically mounted into a volume for the pod to use for Azure API authentication:

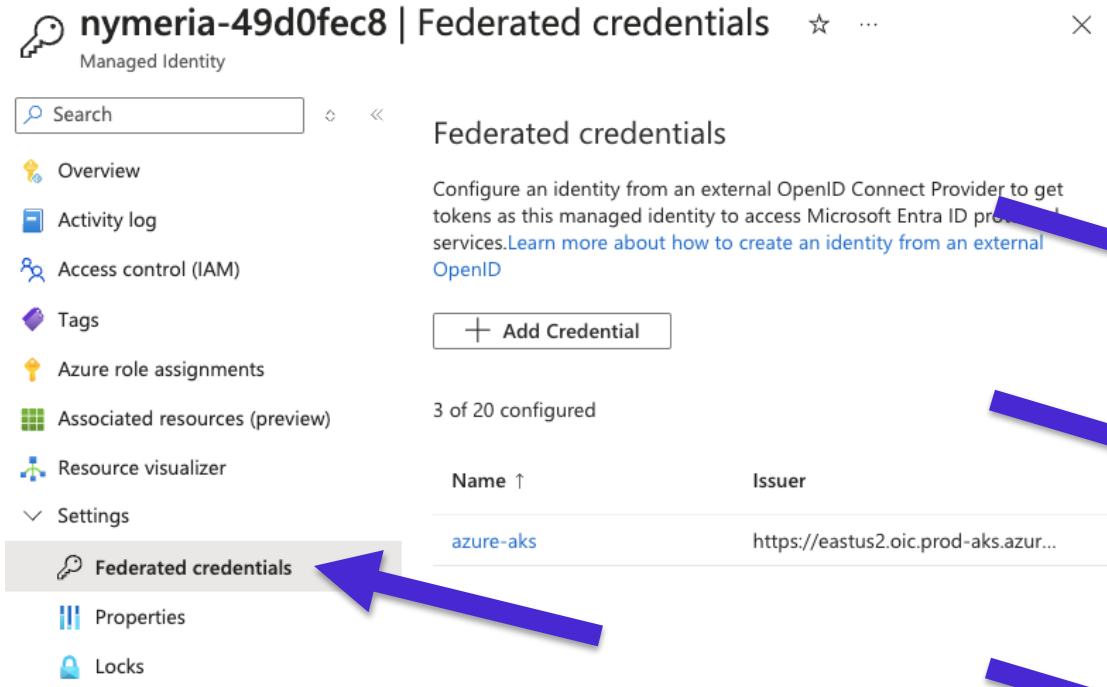
```
$ kubectl exec -n workload-identity -it $(kubectl get pod -n workload-
identity -l cloud=azure -o json | jq -r '.items[0].metadata.name') --
cat /var/run/secrets/azure/tokens/azure-identity-token
```

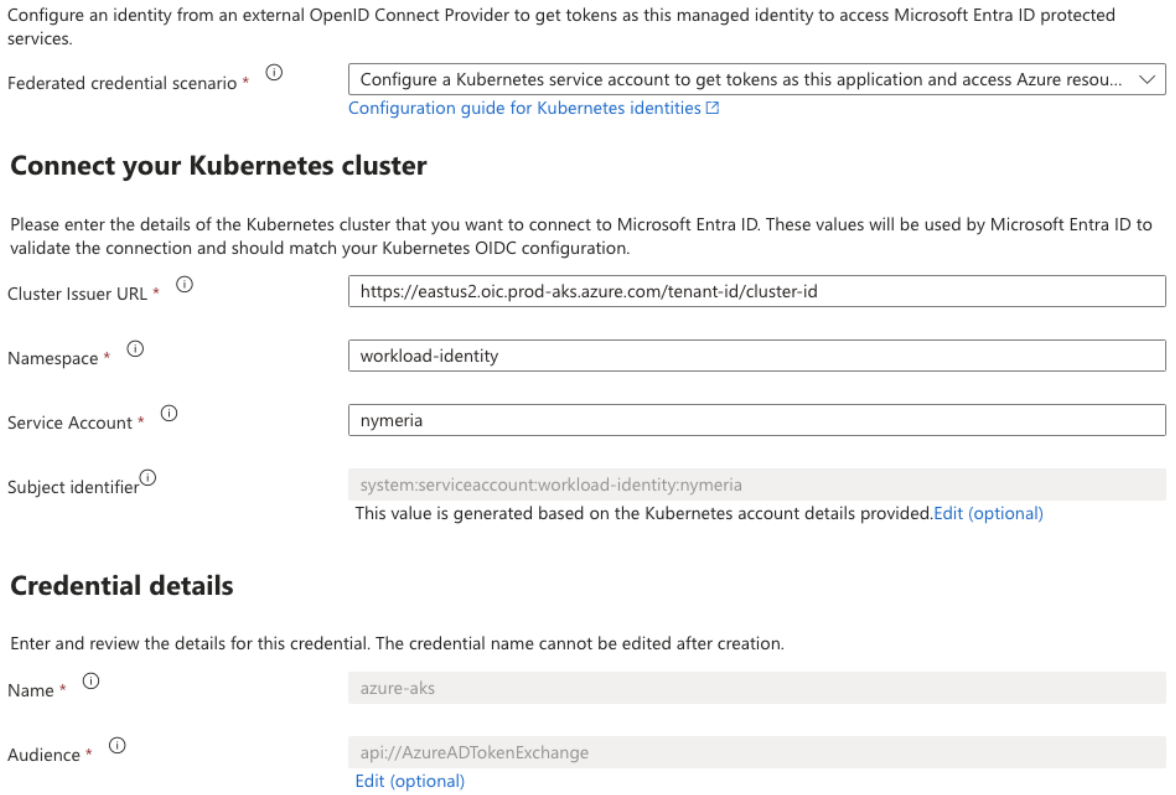
Decoding the service account identity token reveals the AKS cluster's default issuer and audience, as well as the node, pod, and service account:

```
{
  "aud": [ "api://AzureADTokenExchange" ],
  "iss": "https://eastus2.oic.prod-aks.azure.com/tenant-id/cluster-id/",
  "sub": "system:serviceaccount:workload-identity:nymeria"
  "kubernetes.io": {…}
  …
}
```

Azure Managed Identity Federation

1. Azure managed identities support logging in using a federated credential (OIDC JWT)
2. Each federated credential requires the cluster issuer, audience, and service account




Configure an identity from an external OpenID Connect Provider to get tokens as this managed identity to access Microsoft Entra ID protected services.
Federated credential scenario *
Configure a Kubernetes service account to get tokens as this application and access Azure resou...
Configuration guide for Kubernetes identities
Connect your Kubernetes cluster
Please enter the details of the Kubernetes cluster that you want to connect to Microsoft Entra ID. These values will be used by Microsoft Entra ID to validate the connection and should match your Kubernetes OIDC configuration.
Cluster Issuer URL *
https://eastus2.oic.prod-aks.azure.com/tenant-id/cluster-id
Namespace *
workload-identity
Service Account *
nymeria
Subject identifier
system:serviceaccount:workload-identity:nymeria
This value is generated based on the Kubernetes account details provided.Edit (optional)
Credential details
Enter and review the details for this credential. The credential name cannot be edited after creation.
Name *
azure-aks
Audience *
api://AzureADTokenExchange
Edit (optional)

RSAC 2025 Conference

Kubernetes Cross-Cloud Workload Identity Federation

Securing Cloud Access with Kubernetes Workload Identity

Many Voices.
One Community.

Default Identity Token Configuration

The default identity created when enabling workload identity for a cluster are not designed for federating into external APIs:

	Default Audience	Default Expiration
AWS Elastic Kubernetes Service (EKS)	sts.amazonaws.com	1 Day *
Google Kubernetes Engine (GKE)	Cluster Endpoint (self)	1 Year
Azure Kubernetes Service (AKS)	api://AzureADTokenExchange	1 Hour *

* Configurable using the workload identity annotations

Kubernetes Google Projected Volume

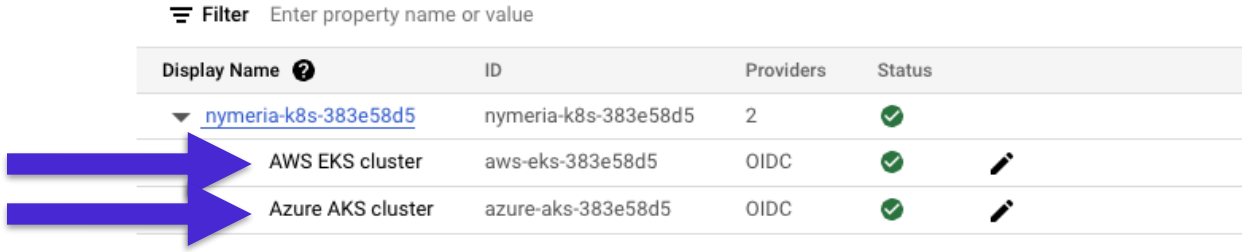
The Google API volume mount configuration for pods running in EKS and AKS:

- Support for *secret*, *configMap*, *serviceAccountToken* data sources
- The *serviceAccountToken* source supports configuration options for setting the token's:
  - Audience (sts.googleapis.com)
  - Expiration (3600 seconds)
- Projected volume is mounted into the containers file system

```
1  volumes:
2    - name: gcp-token
3      projected:
4        sources:
5          - serviceAccountToken:
6              path: token
7              expirationSeconds: 3600
8              audience: "sts.googleapis.com"
9          - configMap:
10           name: gcloud-config
11           items:
12             - key: config.json
13               path: config.json
14 containers:
15   …
16   volumeMounts:
17     - name: gcp-token
18       mountPath: "/var/run/secrets/gcp/
19                   serviceaccount"
```

Google Cloud External Cluster Workload Identity

1. External Kubernetes cluster OIDC providers need to be added to the project's workload identity pool with the issuer and audience:



2. External identities federating through the workload identity provider can be used in direct role bindings:

```
bindings:
  role: roles/storage.objectViewer
  - members:
    - principal://iam.googleapis.com/projects/######/locations/global/
      workloadIdentityPools/nymeria-k8s/
      subject/system:serviceaccount:workload-identity:nymeria
```

Kubernetes AWS Projected Volume

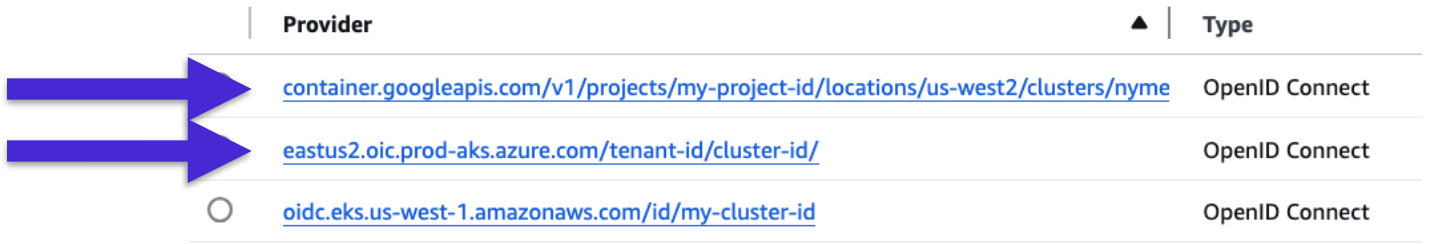
The AWS API volume mount configuration for pods running in AKS and GKE:

- The *serviceAccountToken* source configurates the identity token:
  - Audience (sts.amazonaws.com)
  - Expiration (3600 seconds)
- Projected volume is mounted into the containers file system
- Pod environment variables are set to match the AWS IRSA workload identity configuration

```
1  volumes:
2    - name: aws-token
3      projected:
4        sources:
5          - serviceAccountToken:
6              path: token
7              expirationSeconds: 3600
8              audience: "sts.amazonaws.com"
9  containers:
10   …
11   volumeMounts:
12     - name: aws-token
13       mountPath: "/var/run/secrets/aws/
14                   serviceaccount"
15   env:
16     - name: AWS_ROLE_ARN
17       value:  arn:aws:iam::0123456789012:
18               role/nymeria
19     - name: AWS_WEB_IDENTITY_TOKEN_FILE
20       value: /var/run/secrets/aws/
21              serviceaccount/token
```

AWS Cloud External Cluster Workload Identity

1. External Kubernetes cluster OIDC providers need to be added to the AWS account's identity providers with the issuer and audience:



Provider	Type
container.googleapis.com/v1/projects/my-project-id/locations/us-west2/clusters/nyme	OpenID Connect
eastus2.oic.prod-aks.azure.com/tenant-id/cluster-id/	OpenID Connect
oidc.eks.us-west-1.amazonaws.com/id/my-cluster-id	OpenID Connect

2. Add the external Kubernetes cluster identity provider principals to the IAM Role trust policy to grant external pod service account permissions to assume the role:

```
1  { "Effect": "Allow",
2    "Action": "sts:AssumeRoleWithWebIdentity",
3    "Principal": {
4      "Federated":
5        "arn:aws:iam::123456789012:oidc-provider/eastus2.oic.prod-
6                    aks.azure.com/tenant-id/cluster-id/" },
7  …
```

Kubernetes Azure Projected Volume

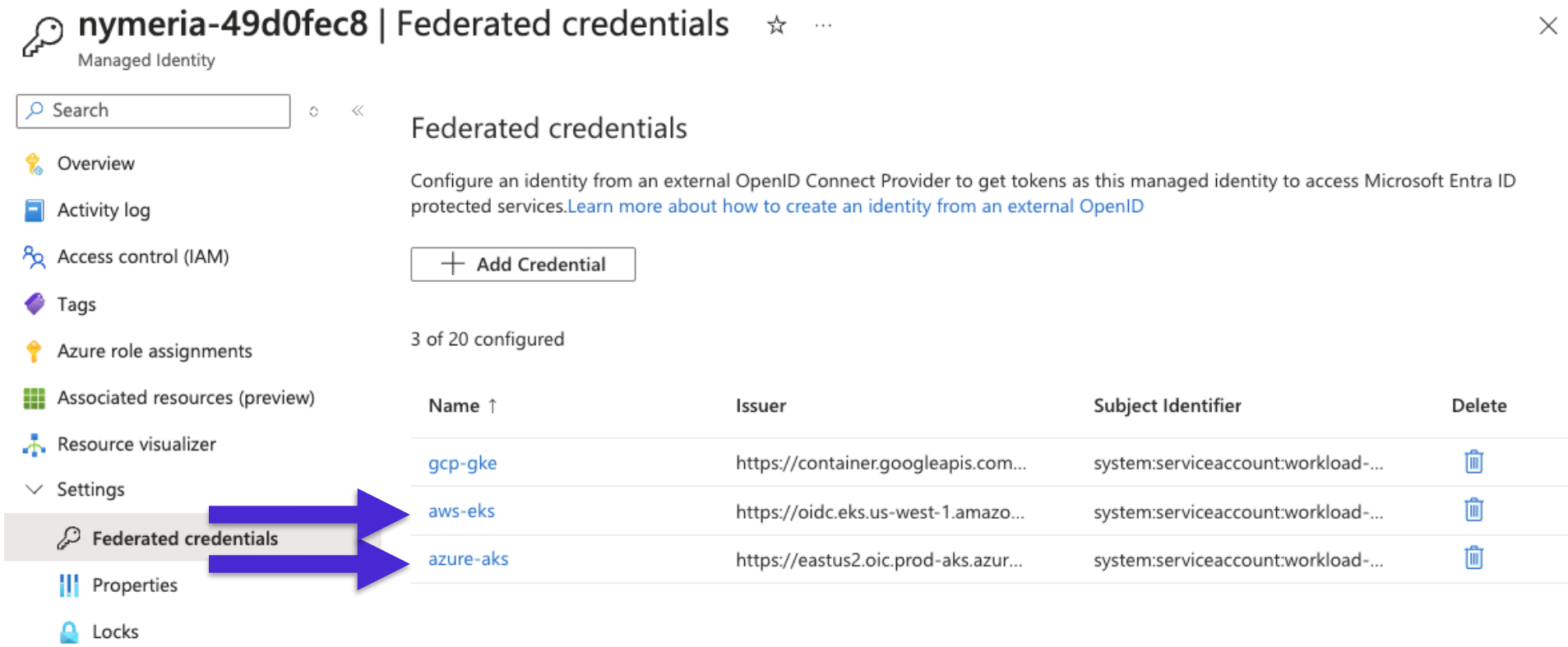
The Azure API volume mount configuration for pods running in EKS and GKE:

- The *serviceAccountToken* source configurates the identity token:
  - Audience (api://AzureADTokenExchange)
  - Expiration (3600 seconds)
- Projected volume is mounted into the containers file system
- Pod environment variables are set to match the Azure workload identity configuration

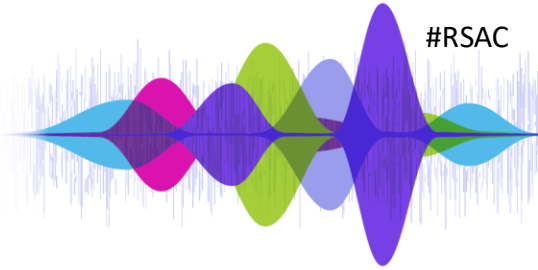
```
1  volumes:
2    - name: azure-token
3      projected:
4        sources:
5          - serviceAccountToken:
6              path: token
7              expirationSeconds: 3600
8              audience:
9                "api://AzureADTokenExchange"
10 containers:
11   …
12   volumeMounts:
13     - name: azure-token
14       mountPath: "/var/run/secrets/azure/
15                   serviceaccount"
16   env:
17     - name: ARM_TENANT_ID …
18     - name: ARM_CLIENT_ID …
19     - name: AZURE_FEDERATED_TOKEN_FILE
20       value: "/var/run/secrets/azure/
21                 serviceaccount/token
```

Azure Cloud External Cluster Workload Identity

External Kubernetes cluster OIDC providers need to be added to the Azure managed identity federated credentials with the issuer and audience:



Apply What You Have Learned Today



- Next week you should:
  - Clone the Nymeria repository and review the documentation
  - Scan your Kubernetes clusters for static credentials stored in *Secrets* and *ConfigMaps*
- Within three months, you should:
  - Configure Kubernetes pods to use intra-cloud integrated workload identity capabilities where possible
- Within six month, you should:
  - Expand Kubernetes workload identity using cross-cloud projected volumes to eliminate cross-cloud static credentials

RSAC 2025 Conference

Many Voices.
One Community.

Thank you for attending!

Eric Johnson
Principal Security Engineer, Puma Security
Senior Instructor, SANS Institute
linkedin.com/in/eric-m-johnson/